

Comparing Web service development with J2EE and Microsoft .NET

Pirjo Prosi

Vaasa Polytechnic

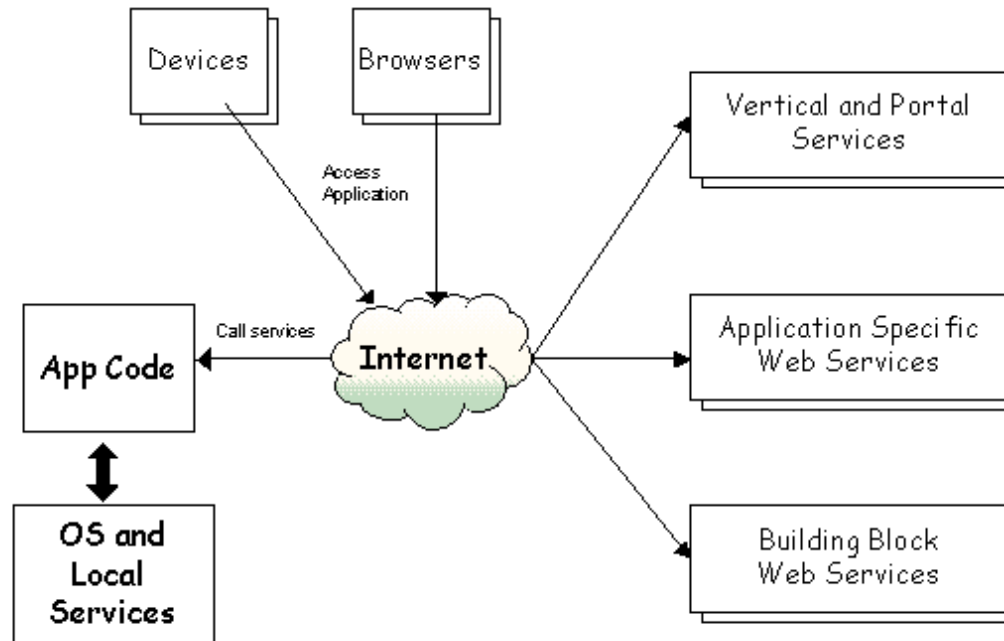
Kimmo Salmenjoki

University of Vaasa

Contents

- **1. Introduction**
- **2. The main technologies of web services**
- **3. Implementing web services**
 - **3.1 Implementing web services with plain SOAP, WSDL and UDDI**
 - **3.2 Implementing web services with Java J2EE**
 - **3.3 Implementing web services with .NET**
- **4. Comparing the web service implementations**
- **5. Conclusions**

1. Introduction



- Internet when using web service architecture [5].

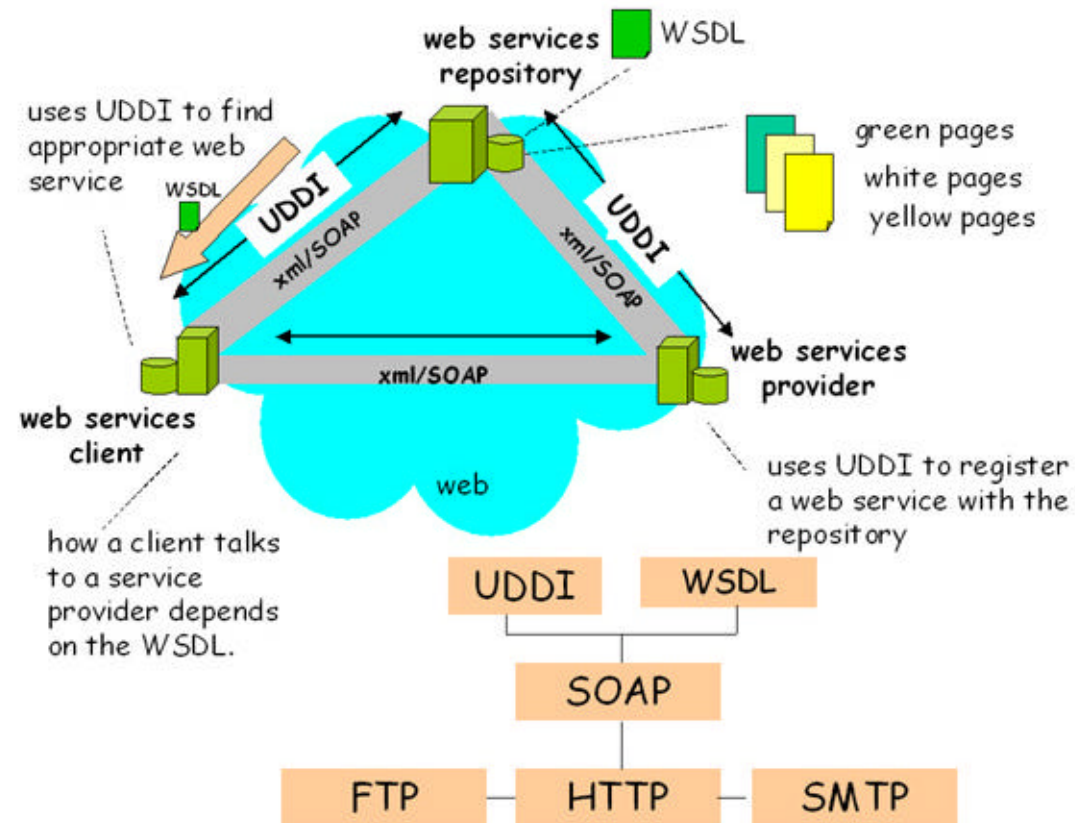
Introduction, cont.

- The main categories of web services implementation activities:
 - Microsoft's .NET framework
 - Application servers (the J2EE community)
 - Integration brokers
 - Database vendors
 - ERP, CRM, and others
 - Web service platforms

2. The main technologies of web services

- XML – Extensible Markup Language describing information
- WSDL – describing Web services
- SOAP – accessing Web services
- UDDI – finding Web services

The main technologies of web services, cont.



- Different roles: web service provider, web service consumer, web service broker

The main technologies of web services, cont.

- The simple object access protocol (SOAP):
 - the most significant of all web services technologies
 - getting the data from one place to another over the network using the standard web transport protocol like HTTP

The main technologies of web services, cont.

- The web services description language (WSDL):
 - describe and publish the formats and protocols of a web service in a standard way
 - The three major elements of WSDL:
 - data types
 - operations
 - binding (defines the transport used)

The main technologies of web services, cont.

- The universal description, discovery, and integration (UDDI) registry:
 - a mechanism for registering and discovering business information over the Internet
- **For web services to be useful and widely used, SOAP, WSDL, and UDDI need to be widely implemented and available in software products.**

The main technologies of web services, cont.

- The inclusion of web services into IT systems and business processes can be achieved with a varying degree of automation:
 - a) Manual invocation of web services via a generic web service client
 - b) Invocation of web services in the context of a traditional application
 - c) Invocation in the context of a custom-designed business process
 - d) (Semi-)automatic localization of web services and (semi-)automatic negotiation of business process. ebXML is an example

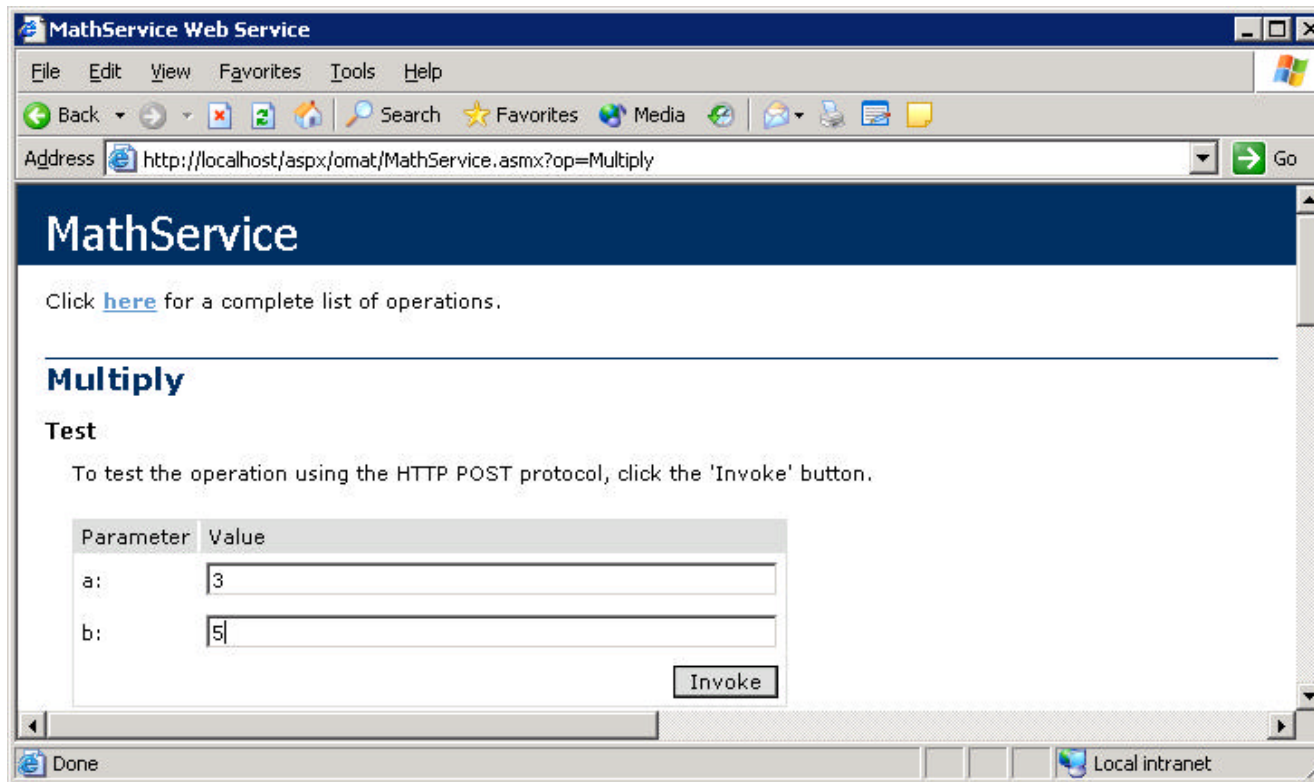
3. Implementing Web services

3.1 Tutorial example implementing web services with plain SOAP, WSDL and

UDDI

- a simple example web service to show the usage of these open standards to describe the web service interaction on the web between the web service provider and its consumer

Tutorial example, cont.



Calling a web service Multiply of the MathService web service in Microsoft ASP.NET Quickstart tutorial, <http://www.asp.net/Tutorials/quickstart.aspx>

Tutorial example, cont.

The following is a sample SOAP request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /webservices/asp/omat/MathService.aspx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/Multiply"

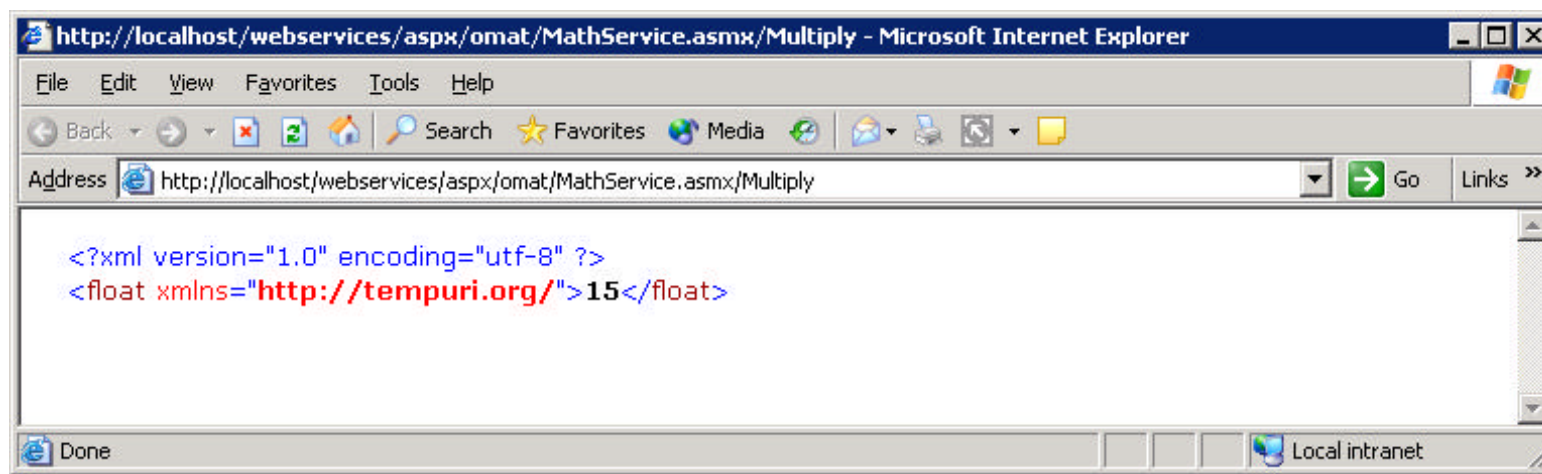
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001
  <soap:Body>
    <Multiply xmlns="http://tempuri.org/">
      <a>float</a>
      <b>float</b>
    </Multiply>
  </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001
  <soap:Body>
    <MultiplyResponse xmlns="http://tempuri.org/">
      <MultiplyResult>float</MultiplyResult>
    </MultiplyResponse>
  </soap:Body>
</soap:Envelope>
```

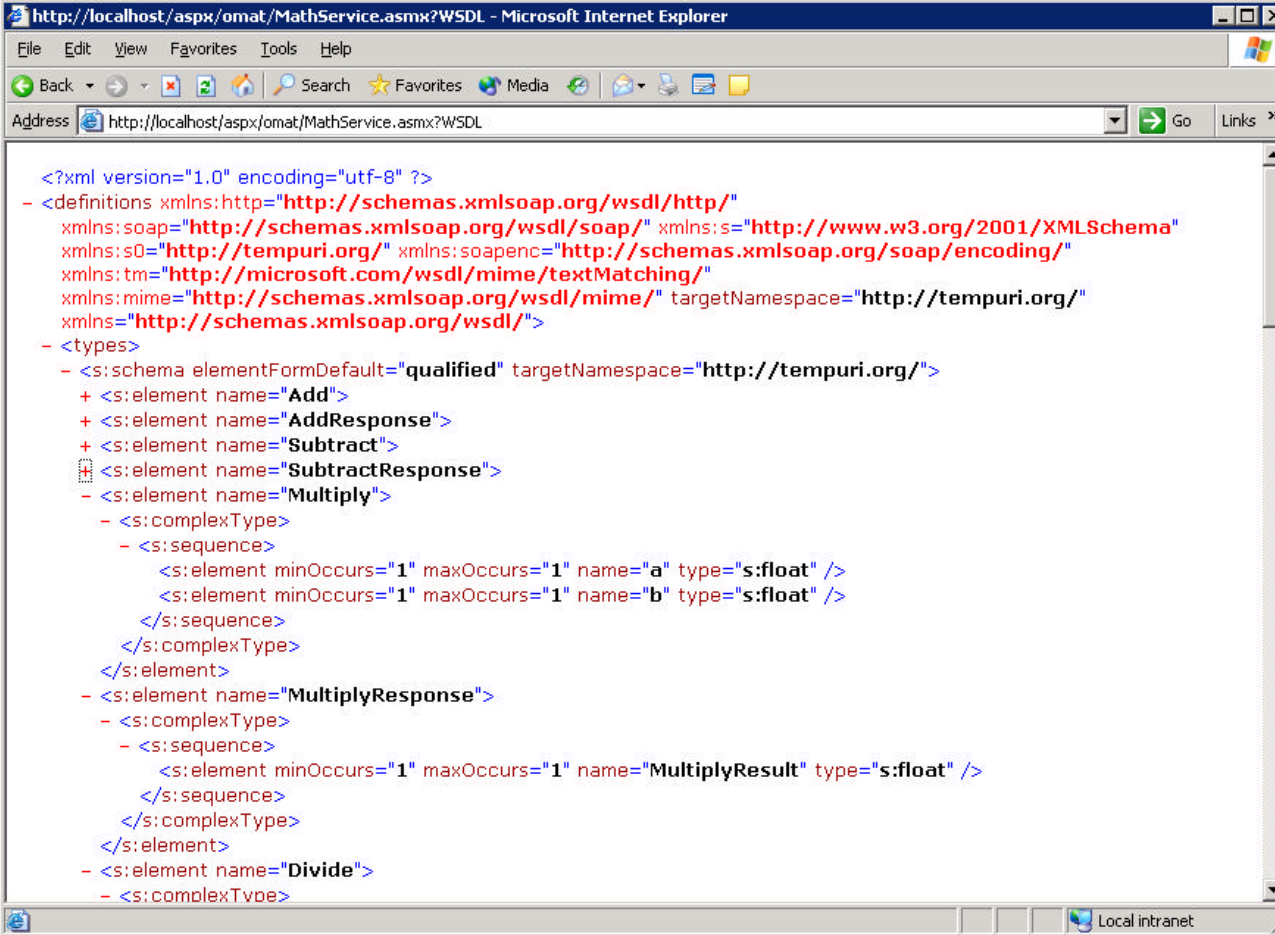
Web service Multiply SOAP messages

Tutorial example, cont.



Web service Multiply SOAP reply messages

Tutorial example, cont.



The screenshot shows a Microsoft Internet Explorer browser window displaying a WSDL (Web Services Description Language) document. The address bar shows the URL: `http://localhost/aspx/omat/MathService.asmx?WSDL`. The main content area displays the XML code of the WSDL, which defines a web service named 'Multiply' with various operations and data types.

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="http://tempuri.org/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
- <types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
+ <s:element name="Add">
+ <s:element name="AddResponse">
+ <s:element name="Subtract">
+ <s:element name="SubtractResponse">
- <s:element name="Multiply">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="a" type="s:float" />
  <s:element minOccurs="1" maxOccurs="1" name="b" type="s:float" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="MultiplyResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="MultiplyResult" type="s:float" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="Divide">
- <s:complexType>
```

Web service Multiply WSDL description

3.2 Implementing web services with the Java 2 Platform, Enterprise Edition (J2EE)

- Why use J2EE when implementing web services?
 - the Java platform makes code portable
 - J2EE gives necessary features: security, distributed transaction management, connection pool management
 - components are reusable
- The J2EE 1.4 specification will contain native support for web services (Beta 2 available Aug. 2003).
- The examples of this paper has been developed using Java Web Services Developer Pack (Java WSDP).

J2EE, cont.

- The Java APIs for XML
 - Java API for XML Processing (JAXP) - processes XML documents using various parsers
 - Java Architecture for XML Binding (JAXB) - processes XML documents using schema-derived JavaBeans component classes
 - Java API for XML-based RPC (JAX-RPC) - sends SOAP method calls to remote parties over the Internet and receives the results
 - Java API for XML Messaging (JAXM) and Soap with Attachments API for Java (SAAJ) - sends SOAP messages over the Internet in a standard way
 - Java API for XML Registries (JAXR) - provides a standard way to access business registries and share information

J2EE: JAX-RPC

- JAX-RPC is the Java API for developing and using web services
- an RPC-based web service is a collection of procedures that can be called by a remote client over the Internet; it is actually a servlet and it is deployed on a server-side container
- uses HTTP, SOAP, and WSDL

J2EE: JAX-RPC example

- A simple user registration service which can be used with different web applications
- using JAX-RPC, the web service itself is basically two files, an interface that declares the service's remote procedures and a class that implements those procedures
- The interface:

```
package users;
```

```
import java.rmi.Remote;
```

```
import java.rmi.RemoteException;
```

```
public interface UsersIF extends Remote {
```

```
    public boolean register(String username, String password,  
String email, String appname) throws RemoteException;
```

```
    public boolean isRegistered(String username, String password,  
String appname) throws RemoteException;
```

```
    //other remote procedures
```

```
}
```

J2EE: JAX-RPC example, cont.

The outline of the implementation class:

```
package users;

public class UsersImpl implements UsersIF {

    public boolean register(String username, String password,
String email, String appname) {
        //This method saves username, password and email in
        //database. The database table depends on the name of the
        //application (appname). If registration fails - for
        //example, if there already exists the same username -
        //this method returns boolean value false.
    }

    public boolean isRegistered(String username, String password,
String appname) {
        //This method checks if the given username and password
        //pair exists in the database.
    }

    //Other methods
}
```

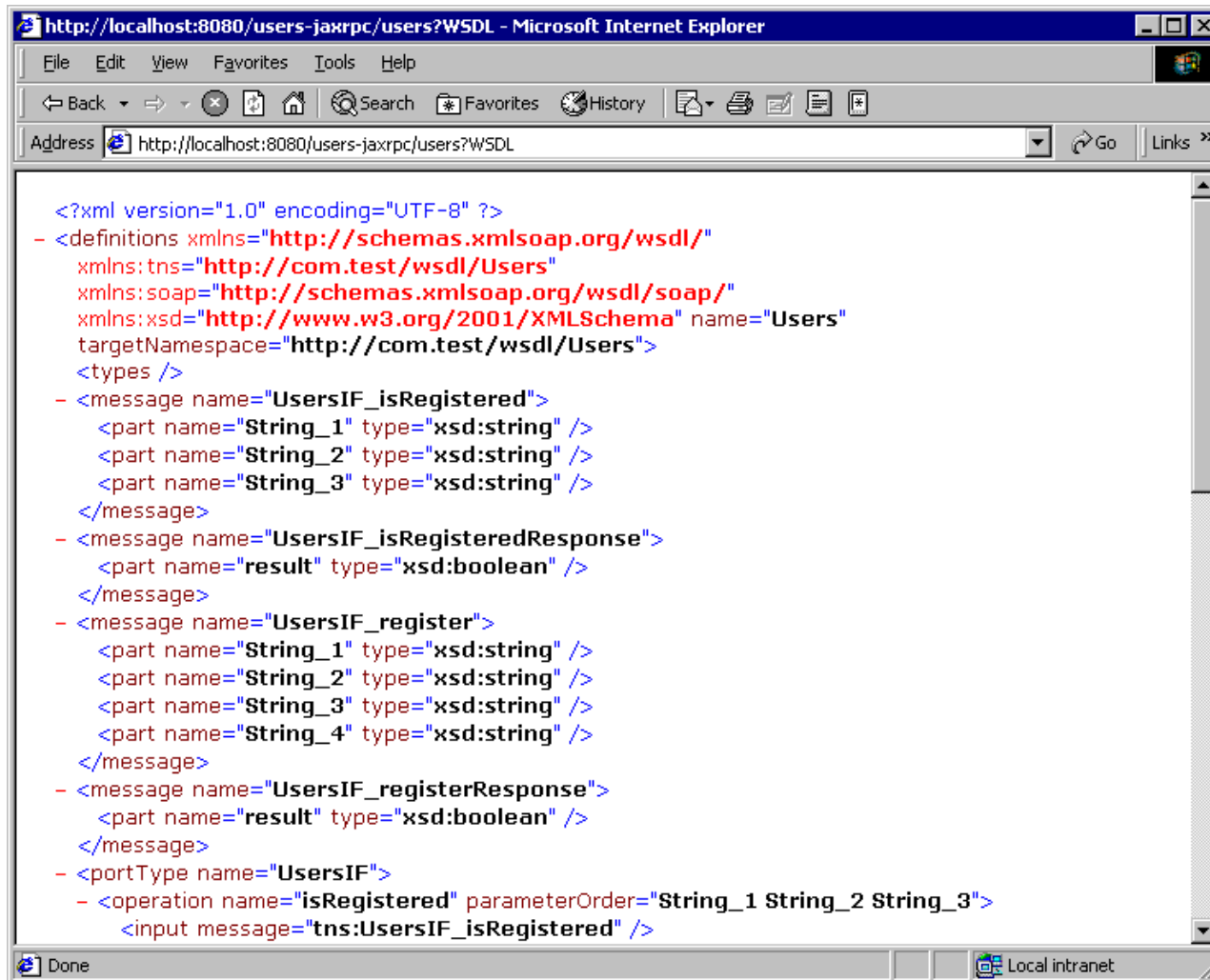
J2EE: JAX-RPC example, cont.

- next step is to configure and run the mapping tool:
 - it uses the interface and its implementation as a basis for generating the **stub** and **tie** classes plus other classes as necessary
 - it creates the WSDL description for the service
- the final steps in creating a web service are packaging and deployment

Web Services

Port Name	Status	Information
Users	ACTIVE	Address: http://localhost:8080/users-jaxrpc/users WSDL: http://localhost:8080/users-jaxrpc/users?WSDL Port QName: {http://com.test/wsdl/Users} UsersIFPort Remote interface: users.UsersIF Implementation class: users.UsersImpl Model: http://localhost:8080/users-jaxrpc/users?model

J2EE: JAX-RPC example, cont.



The screenshot shows a Microsoft Internet Explorer browser window with the address bar containing `http://localhost:8080/users-jaxrpc/users?WSDL`. The main content area displays the following XML WSDL document:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://com.test/wsdl/Users"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="Users"
  targetNamespace="http://com.test/wsdl/Users">
  <types />
  - <message name="UsersIF_isRegistered">
    <part name="String_1" type="xsd:string" />
    <part name="String_2" type="xsd:string" />
    <part name="String_3" type="xsd:string" />
  </message>
  - <message name="UsersIF_isRegisteredResponse">
    <part name="result" type="xsd:boolean" />
  </message>
  - <message name="UsersIF_register">
    <part name="String_1" type="xsd:string" />
    <part name="String_2" type="xsd:string" />
    <part name="String_3" type="xsd:string" />
    <part name="String_4" type="xsd:string" />
  </message>
  - <message name="UsersIF_registerResponse">
    <part name="result" type="xsd:boolean" />
  </message>
  - <portType name="UsersIF">
    - <operation name="isRegistered" parameterOrder="String_1 String_2 String_3">
      <input message="tns:UsersIF_isRegistered" />
```

The WSDL-document of the users registration service (on browser).

J2EE: JAX-RPC example, cont.

- writing the client application:
 - writing code that invokes the desired method
 - a client creates a proxy, a local object representing the service, and then simply invokes methods on the proxy
 - the WSDL document of the web service is a basis for creating the proxy
 - a JAX-RPC client can use next programming models to invoke a web service endpoint:
 - the stubs-based model: the stub is created before runtime (a static stub)
 - the dynamic proxy model: the proxy is created during runtime
 - the dynamic invocation interface DII: a client can call a remote procedure even if the signature of the remote procedure or the name of the service are unknown until runtime

J2EE: JAX-RPC example, cont.

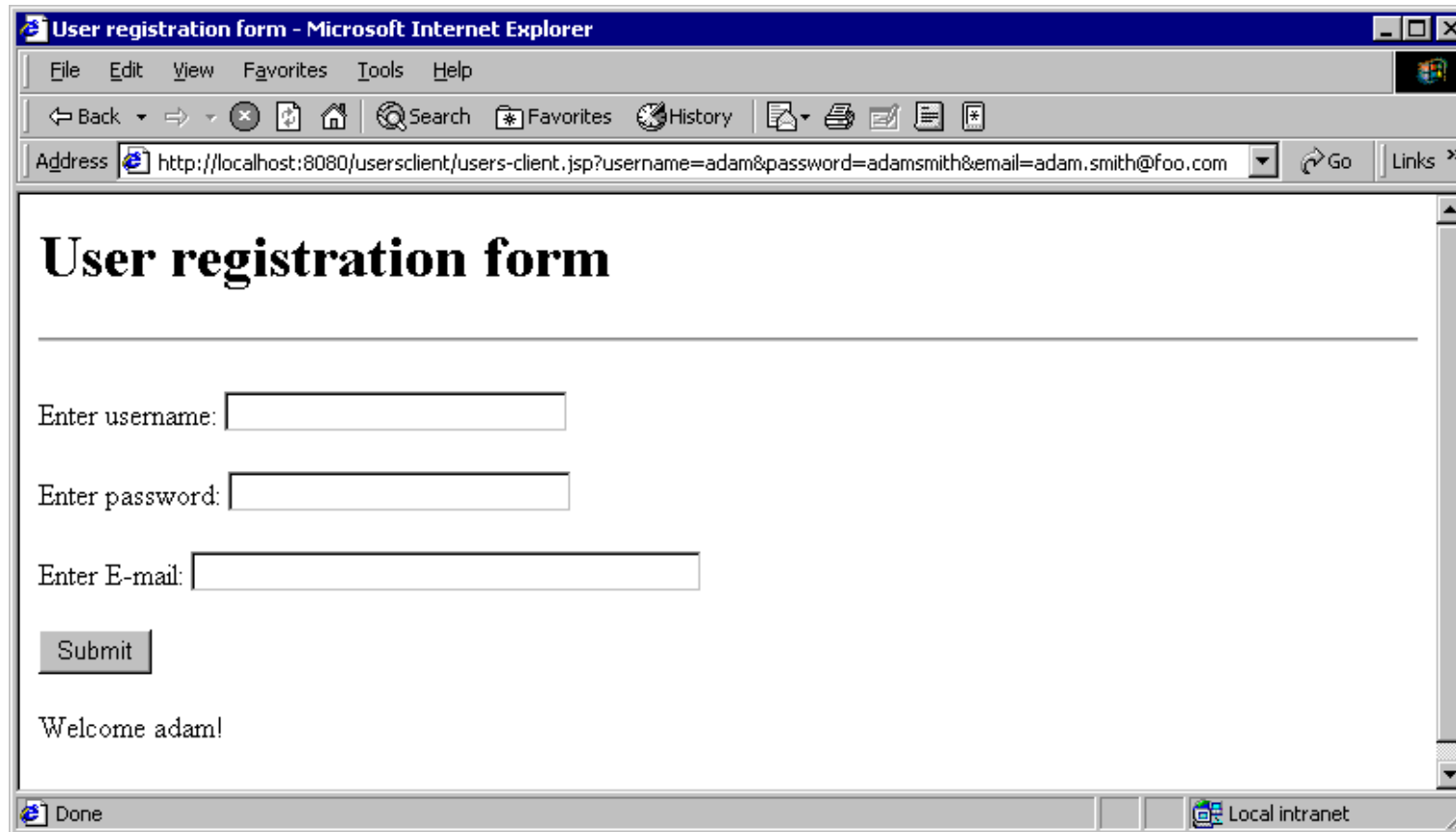
- a dynamic proxy client, which calls the web service's **registerate**-operation

```
// The necessary information for finding the service
String urlString = "http://localhost:8080/users-jaxrpc/users?WSDL";
String namespaceUri = "http://com.test/wsdl/Users";
String serviceName = "Users";
String portName = "UsersIFPort";
ServiceFactory serviceFactory = ServiceFactory.newInstance();
Service usersService =
    serviceFactory.createService(new URL(urlString),
        new QName(namespaceUri, serviceName));
//Creating the proxy
users.UsersIF myProxy =
    (users.UsersIF) usersService.getPort(new
        QName(namespaceUri, portName),
        users.UsersIF.class);
//Invoking the remote operation
boolean result = myProxy.register(username, password, email,
    "user");
```


J2EE: JAX-RPC example, cont.

- a JAX-RPC runtime system:
 - converts the client's remote method call into a SOAP message and sends it to the service as an HTTP request
 - on the server side, the JAX-RPC runtime system receives the request, translates the SOAP message into a method call, and invokes it
 - after the web service has processed the request, the runtime system goes through a similar set of steps to return the result to the client
 - as complex as the implementation details of communication between the client and server may be, they are invisible to both web services and their clients.

J2EE: JAX-RPC example, cont.



Using the JSP-document and invoking the web service.

J2EE: JAX-RPC example, cont.

- a client program invoking the web service with DII; this can be used in the web service broker (type d web service usage, see chapter 2)

```
ServiceFactory serviceFactory = ServiceFactory.newInstance();
Service usersService =
    serviceFactory.createService(new QName("Users"));
QName port = new QName("UsersIF");
Call call = usersService.createCall(port);
call.setTargetEndpointAddress("http://localhost:8080/users-
jaxrpc/users");
QName QNAME_TYPE_BOOLEAN = new QName(NS_XSD, "boolean");
QName QNAME_TYPE_STRING = new QName(NS_XSD, "string");
call.setReturnType(QNAME_TYPE_BOOLEAN);
call.setOperationName(new QName("http://com.test/wsdl/Users",
"isRegistered"));
Object[] params = new Object[3];
call.addParameter("String_1", QNAME_TYPE_STRING, ParameterMode.IN);
...
//Invoking the remote operation
boolean result = ((Boolean)call.invoke(params)).booleanValue();
```

3.3. Implementing web services with .NET

Web services	Web Forms	Windows Forms
Data and XML classes		
Base Classes		
Common Language Runtime (CLR)		

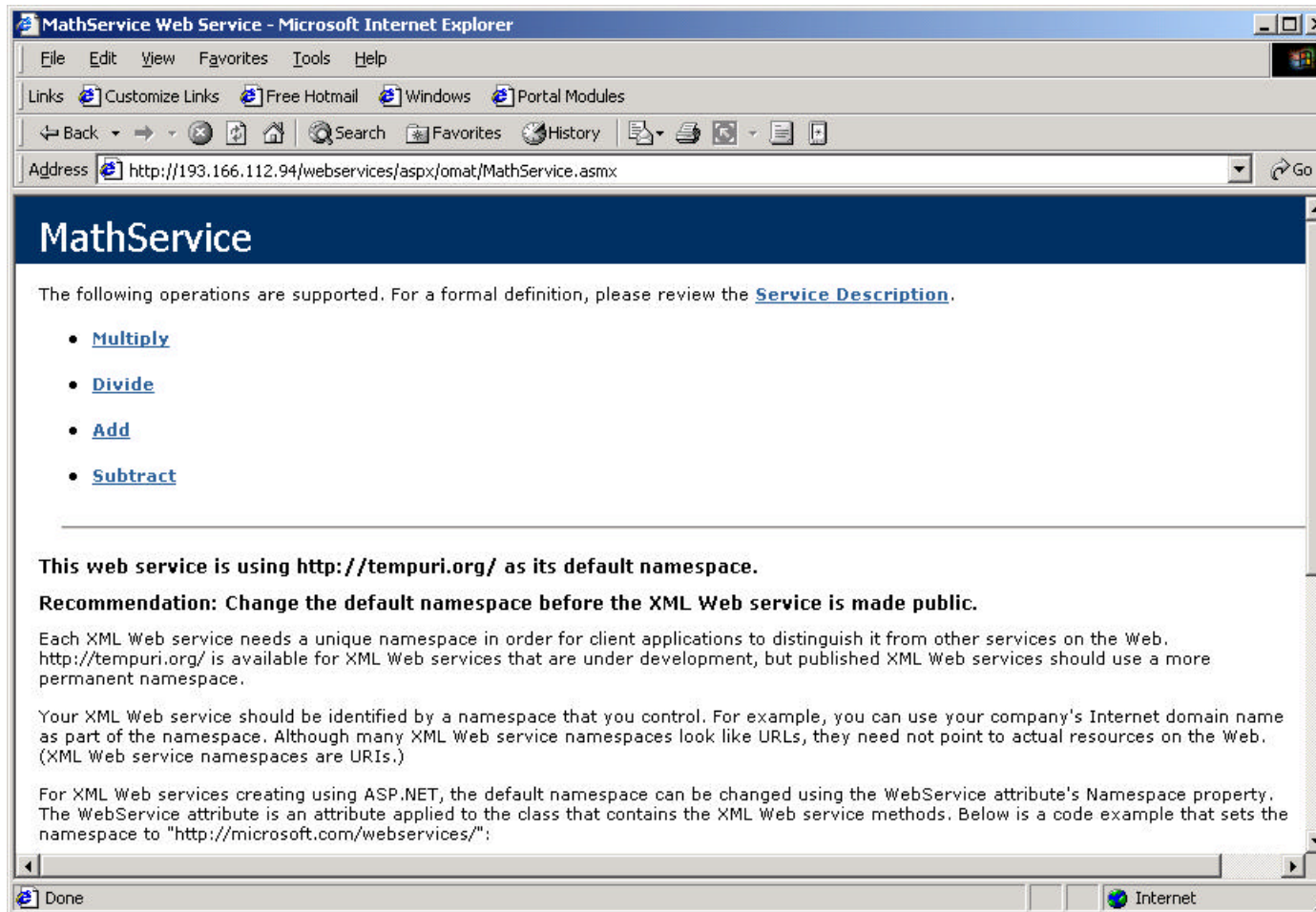
MS .NET internal architecture, see

<http://msdn.microsoft.com/netframework/technologyinfo/overview/>

.NET, cont.

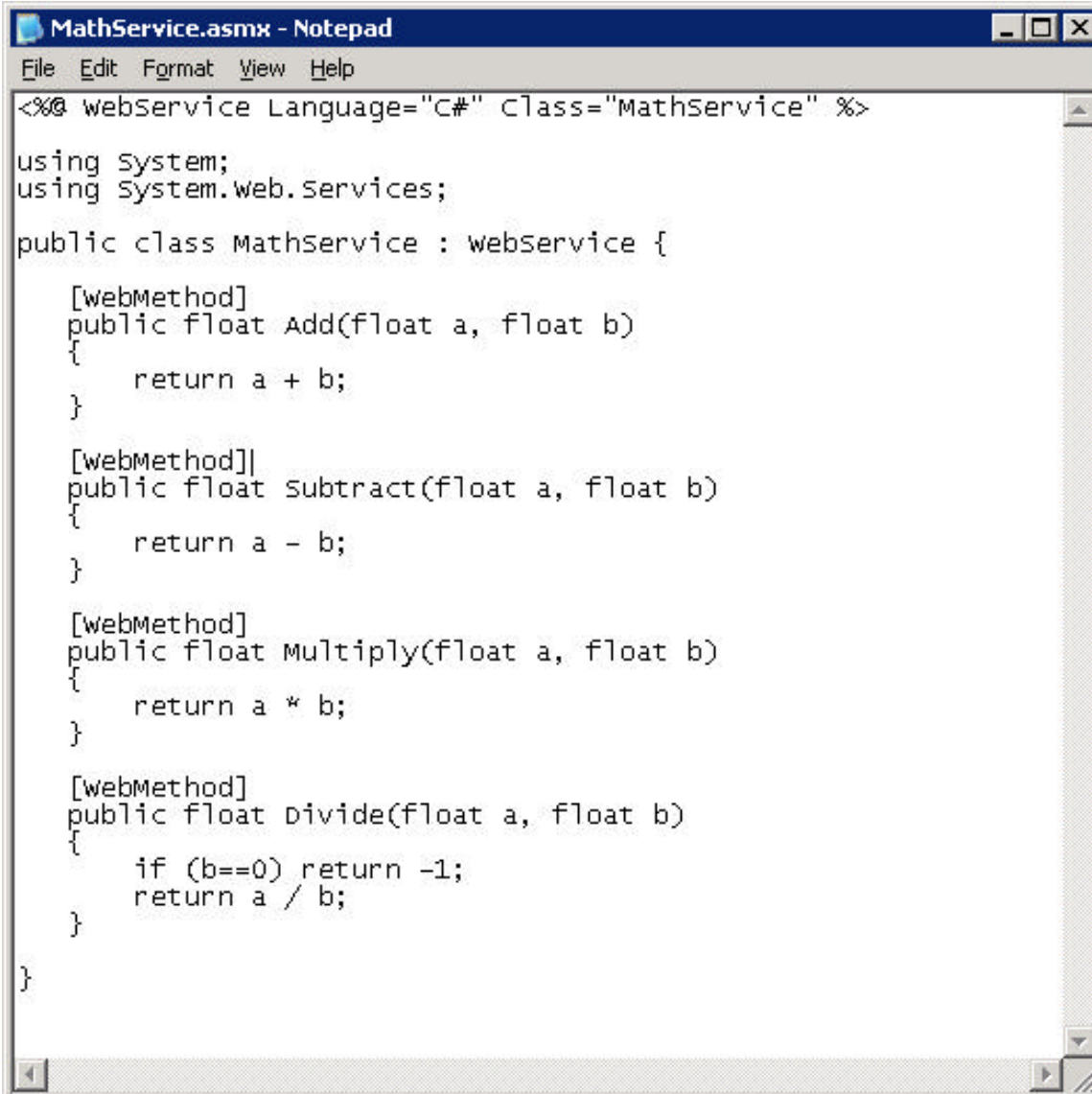
- ASP.NET:
 - an enhanced version of Microsoft's Active Server Pages, which offers a new mechanism for exposing web-oriented information
 - in addition to extended services through a new file type (.asmx) and the ability to compile your ASP.NET pages into DLLs (versus simple script execution), ASP.NET offers you attribute-based programming services.

.NET, cont.



Example web service description and documentation

.NET, cont.



```
MathService.asmx - Notepad
File Edit Format View Help
<%@ webservice Language="C#" Class="MathService" %>

using System;
using System.Web.Services;

public class MathService : webservice {

    [webMethod]
    public float Add(float a, float b)
    {
        return a + b;
    }

    [webMethod]
    public float Subtract(float a, float b)
    {
        return a - b;
    }

    [webMethod]
    public float Multiply(float a, float b)
    {
        return a * b;
    }

    [webMethod]
    public float Divide(float a, float b)
    {
        if (b==0) return -1;
        return a / b;
    }
}
```

- **Source code of the web service MathService.asmx**

.NET, cont.

- using .NET, all of the previously shown SOAP, WSDL (and UDDI) files are generated automatically from this web service code file `MathService.asmx`
- these files can also be coded in various languages (C#, Visual Basic, J#, and Java Script)
- also the execution of the services does not require any XML editing, but one is directly using the appropriate access type (and environment) for the service

.NET, cont.

- .NET is a Microsoft-centric approach to web services
- .NET runs on a single platform (Windows) but it does natively support multiple languages
- J2EE, on the other hand, is a platform-independent solution deployed in a single language (Java)
- The Global XML Web Services Architecture by Microsoft is built using the XML and web services. Its architecture is divided into three conceptual layers: SOAP, SOAP modules, and infrastructure protocols.

3.4 Combining J2EE and .NET web service example

- Web service consumer and provider exchange SOAP messages over HTTP
- the implementational details of the respective systems require that the WSDL description of the service is compiled to a supporting file (at appropriate language and format) recognized by the environment

4. Comparing the web service implementations

4.1 Web service and software construction

- implementation efforts tend to fall into two major categories leaving web services developers and deployers with a choice: the Microsoft environment and the Java environment
- environments are compatible, at least to the extent that they both implement the same specifications and interpret them the same way

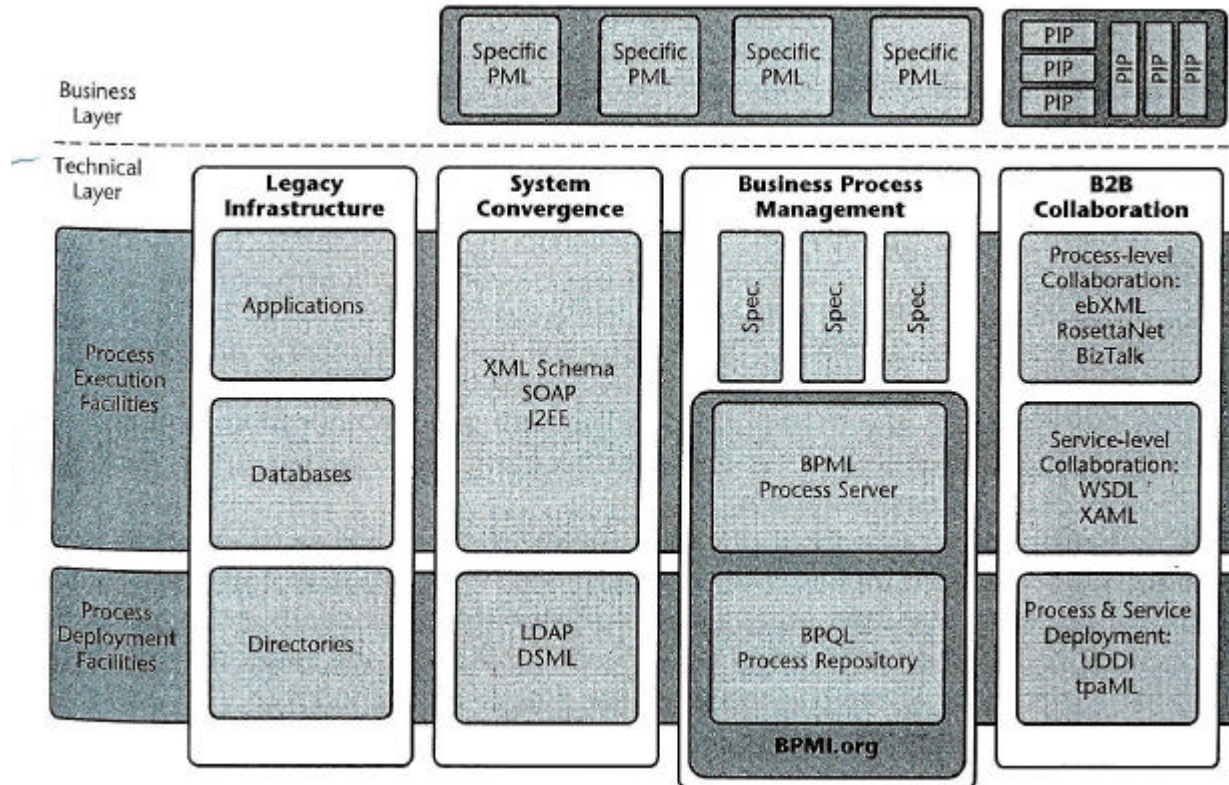
Web service and software construction, cont.

- J2EE vs. .NET decision is more about in which application development environment web services will be deployed:
- Today's situation: .NET will probably become the de facto standard for client-side application development and J2EE for server-side
- The deployment platform: in Windows .NET, in heterogeneous environments J2EE
- The economic and philosophical issues: Microsoft's monopolistic practices, licensing costs and product quality, the attraction to Microsoft's solutions, etc.

Web service and software construction, cont.

- with the .NET approach, development of a web service occur in a single stage: once you write a program in any of the .NET languages, you have only to push a button to create and deploy it as a web service
- every program is a web service to .NET, or at least potentially so
- in the Java world, web services are a two-step process in with programmers develop classes and beans and then decide as a second step which of them, if any, are to be created and deployed as web services.

4.2 Web services in electronic business



The overall BPMI (Business Process Management Initiative) specification deal with processes in electronic business. They aim at process integration in intra- and inter-enterprise computing.

Web services in electronic business, cont.

- web services can
 - run on desktop and handheld clients to access such Internet applications as reservations systems and order-tracking systems
 - be used for business-to-business (B2B) integration, connecting applications run by various organizations in the same supply chain
 - solve the broader problem of enterprise application integration (EAI), connecting multiple applications from a single organization to multiple other applications both inside and outside the firewall
- in all these cases, the technologies of web services provide the standard glue connecting diverse pieces of software

Web services in electronic business, cont.

- Business-to-business communication requires infrastructure and intermediaries
- Internet B2B intermediaries provide transparency regarding price, product features, availability, and supplier characteristics
- Typical examples of intermediaries: VerticalNet, Chemdex (now Ventro), Commerce One, and Ariba.
- Global Net Exchange is an initiative by Sears, Carrefour, Sainsbury and others; the competing Worldwide Retail Exchange is an initiative by Kmart, Target, Walgreens, Tesco, Auchan, Casino and others; and the Grocery Manufacturer' Association eCFG is an initiative by P&G, Unilever, Kraft Foods, and others.

Web services in electronic business, cont.

- software engineering web services require techniques for
 - management, scalability, security, and auditing of business process collaborations
- for web service brokering one could also use semantic maps
- web brokers can offer auxiliary services:
 - gather statistics (which services are wanted most, and by whom)
 - arrange for payment of services
 - forward requests to other brokers
- Web services will rely on P2P communication and on sophisticated semantic navigation techniques that allow consumers to find the right service. Or will the services begin to hunt the web for possible consumers?

5. Conclusions

- that web service imply a major paradigm shift for web based software development.
- the standards and technologies behind web services are still under constant development
- all the major IT vendors see web service as a strategic choice for software development
- what ever is your preferred development environment and language it seems that web services are viable option for development of new, web enabled and integrated software components

Conclusions, cont.

- development and deployment of web services requires no specific underlying technology platform; however, the basic choice remains between Microsoft's .NET framework and the Java Community Process' (JCP) J2EE specification
- the realization of web services as a common communications infrastructure requires cooperation between the J2EE and .NET factions
- in short, .NET is a thorough, fundamental rearchitecting of a distributed computing platform based on web services, while application server support for web services tends to be designed more as another client, or presentation tier for the back-end system

References

- 1. E. Armstrong, S. Bodoff, D. Carson, Maydene Fisher, Scott Fordin, Dale Green, Kim Haase, Eric Jendrock: The Java Web Services Tutorial, (Feb. 2003)
<http://java.sun.com/webservices/docs/1.1/tutorial/doc/>
- 2. F. P. Coyle: XML, Web Services and the Changing Face of Distributed Computing, ACM Ubiquity magazine online: http://www.acm.org/ubiquity/views/f_coyle_1.html
- 3. B. Daum, U. Merten: System Architecture with XML, Morgan Kaufmann, 2003
- 4. C. F. Goldfarb: [XML Handbook](#), 4th edition, 2002, Barnes&Noble
- 5. M. Kirtland: Platform for Web Services by, Microsoft Developer Network, January 2001
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/websvcs_platform.asp

References, cont.

- 6. Microsoft Inc..NET, <http://www.microsoft.com/net/>
- 7. G. Miller: The Web Services Debate. .NET vs. J2EE, *Communications of the ACM*, June 2003, Vol. 46, No. 6, p. 64-67
- 8. E. Newcomer: Understanding Web Services XML, WSDL, SOAP, and UDDI, Addison-Wesley, 2002
- 9. Sun Inc.
- 10. J. Williams: The Web Services Debate. J2EE vs. .NET, *Communications of the ACM*, June 2003, Vol. 46, No. 6, p. 59-63
- 11. The SOAPBuilders Interoperability Lab, <http://www.xmethods.net/ilab/>